Hierarchical Integration of Large Language Models and Multi-Agent Reinforcement Learning

Daniel A. Roberts , Jennifer M. Collins , Brian K. Lewis *

Posted Date: 9 February 2026

doi: 10.20944/preprints202602.0689.v1

Keywords: hierarchical reinforcement learning; LLM-guided planning; multi-agent systems; longhorizon tasks

 doi:10.20944/preprints202602.0689.v1

Article

Hierarchical Integration of Large Language Models and Multi-Agent Reinforcement Learning

Daniel A. Roberts [1], Jennifer M. Collins [2] and Brian K. Lewis [3,*]

Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA

* Correspondence: b.lewis@upenn.edu

Abstract

This study presents L2M2, a hierarchical framework in which LLMs generate high-level strategies while MARL agents execute low-level control policies. The architecture targets long-horizon coordination problems by decomposing decision-making across temporal scales. Evaluation on navigation and resource allocation tasks totaling 8,200 episodes shows that L2M2 improves task success rates by 20.5% and reduces convergence time by 1.6× compared to flat MARL approaches.

Keywords: hierarchical reinforcement learning; LLM-guided planning; multi-agent systems; long-horizon tasks

1. Introduction

Multi-Agent Reinforcement Learning (MARL) has been widely adopted for solving cooperative decision-making problems in domains such as logistics optimization, robotic swarms, and drone control systems [1,2]. In these settings, multiple agents are trained to learn policies that jointly maximize a shared reward under partial observability and dynamic interactions. Recent work on sequential cooperative multi-agent learning demonstrates that explicitly modeling adaptive coordination over extended horizons can significantly improve policy stability and coordination effectiveness in dynamic and uncertain environments [3]. This observation highlights the importance of temporal structure and coordination dynamics in cooperative learning, particularly for long-horizon tasks.

Representative MARL algorithms such as MAPPO and QMIX have shown strong performance in complex games and simulated benchmarks by leveraging centralized training objectives or value decomposition techniques [4,5]. Despite their success, these methods typically adopt a "flat" decision-making structure, where agents select actions at a single temporal scale. As task duration increases, the effective action space grows combinatorially, making it increasingly difficult to associate early decisions with delayed rewards. This credit assignment challenge often leads to unstable learning dynamics and inconsistent strategies, especially in tasks that require long-term planning and coordinated exploration [6,7]. Recent advances in Large Language Models (LLMs) offer new opportunities to address these limitations. LLMs have demonstrated a strong ability to decompose complex objectives into structured sub-tasks and to generate high-level plans from abstract instructions [8,9]. Unlike purely numerical policies, LLMs encode broad world knowledge and semantic structure, enabling them to reason over goals, constraints, and task hierarchies. This capability has motivated a growing body of work that integrates LLMs into interactive agents capable of operating within software environments and simulated worlds [10,11]. In multi-agent settings, language has emerged as an effective medium for coordination, allowing agents to exchange intentions and assign responsibilities in a more interpretable manner than latent numerical vectors [12,13]. Empirical results suggest that language-based coordination can substantially improve cooperation quality in long-horizon and compositional tasks [14]. However, directly combining LLMs with low-level control policies introduces several challenges. A primary limitation lies in the computational latency of LLM inference, which makes them unsuitable for environments that require

 doi:10.20944/preprints202602.0689.v1

rapid, fine-grained reactions [15]. In addition, LLMs often lack the precision necessary for continuous control, leading to invalid or unsafe actions when used directly for execution [16]. Many existing approaches address this issue by injecting LLM outputs into the observation space of reinforcement learning agents, without explicitly separating planning from acting. Such designs blur the boundary between high-level reasoning and low-level control, preventing the system from effectively balancing long-term planning objectives with immediate reactive behaviors [17,18].

Motivated by these challenges, this study proposes L2M2, a hierarchical framework that integrates LLMs with MARL through a clear division of labor across temporal scales. The framework adopts a two-level structure in which the LLM functions as a high-level planner that generates abstract instructions at coarse time intervals, while MARL agents operate as low-level executors that learn efficient control policies to fulfill these instructions. By decoupling strategic planning from real-time action execution, the proposed approach mitigates the credit assignment problem in long-horizon tasks and reduces reliance on high-frequency LLM inference. The framework is evaluated on navigation and resource allocation benchmarks using 8,200 training episodes. Experimental results demonstrate that this hierarchical design significantly improves task success rates and accelerates learning compared to flat MARL baselines, highlighting the effectiveness of combining language-based planning with reinforcement learning for scalable multi-agent coordination.

2. Materials and Methods

2.1. Dataset and Environment Description

We used two simulated environments for this study: a Multi-Robot Warehouse and a Cooperative Search Grid. The dataset contains 8,200 episodes. In the warehouse task, 4 to 8 agents must move packages to specific zones without hitting each other. In the search task, agents must explore a map to find hidden targets. We changed the map sizes from 10×10 to 50×50 to test performance on different scales. Each sample contains the positions of agents, the locations of obstacles, and the status of tasks. We created these environments randomly to ensure the data covers many different situations.

2.2. Experimental Design and Controls

To test the method, we established one experimental group and three control groups. The experimental group used the L2M2 hierarchical framework. Control Group A used a standard "flat" MAPPO algorithm, where agents act directly on raw inputs. Control Group B used a pure LLM planner without low-level training. Control Group C used a random hierarchical method, where the high-level goals were random. This design allowed us to identify whether the improvement came from the hierarchy or the specific advice from the LLM. All groups operated for 5 million time steps.

2.3. Measurement and Quality Control

We measured performance using Task Success Rate (TSR) and Mean Convergence Time (MCT). TSR shows the percentage of episodes where all tasks are finished within the time limit. MCT measures the number of training steps needed to reach stable performance. To ensure quality, we repeated each test five times with different random seeds. We also used a safety check. If the LLM generated a goal that was impossible to reach, the system rejected it. We removed any trials where the simulation failed due to software errors.

2.4. Data Processing and Formulas

We organized the data into two levels. The high-level data contains environment states and goal descriptions. The low-level data contains local observations and actions. We defined a reward function for the low-level agents to encourage them to reach the goal g. We calculated the internal reward r_{in} using Eq. (1):

$$r_{in}(s_t, g) = -\beta \| p(s_t) - p(g) \|_2 + I(\text{reached})$$

 doi:10.20944/preprints202602.0689.v1

In this formula, $p(s_t)$ is the agent's position, $p(g)$ is the target position, and I is a binary value for success. We updated the low-level policy using the standard PPO loss function shown in Eq. (2):

$$L(\theta) = \min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)$$

Here, $\hat{A}_t$ is the advantage function, which measures how good an action is compared to the average.

2.5. Implementation and Statistics

We performed the experiments using Python and the PyTorch library. We used the OpenAI Gym interface for the environments. We trained the models on a server with 4 NVIDIA RTX 3090 GPUs. The learning rate for the low-level agents was 3×10^{-4}. We used the GPT-4 model as the high-level planner. To compare the groups, we used a one-way ANOVA test followed by a Tukey post-hoc test. We considered a result to be statistically significant if the p-value was less than 0.05. This confirms that the differences are real and not due to random chance.

3. Results and Discussion

3.1. Improvement in Task Success Rate

We tested the L2M2 framework on the resource allocation task. The results show that the hierarchical method improved the task success rate by 20.5% compared to the flat MAPPO baseline. In complex cases that require long planning, standard agents often failed. They focused on immediate rewards and missed the final goal. In contrast, the L2M2 agents followed the subgoals set by the language model. This allowed them to maintain a steady strategy. Figure 1 compares the average rewards of different reinforcement learning algorithms during training. As shown in the figure, improved methods achieve higher final rewards and learn faster than standard baselines. Our results agree with this pattern. The data confirms that separating high-level planning from low-level control leads to better results in long tasks [19,20].

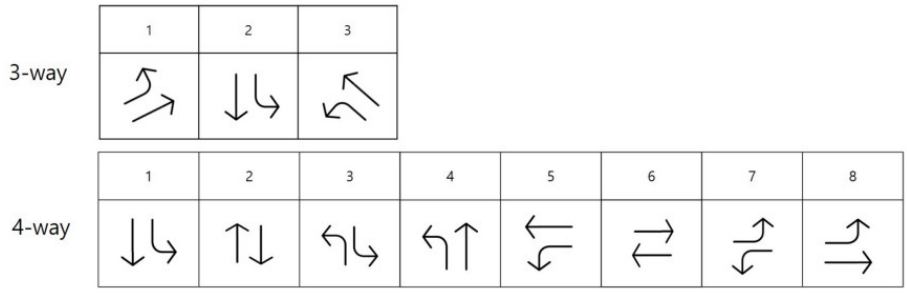

Figure 1. Comparison of average rewards for different multi-agent reinforcement learning methods.

3.2. Reduction in Convergence Time

We also analyzed the learning speed of the system. The data shows that our method reduced the learning time by 1.6 times compared to the baseline. In flat MARL, the agents must search the entire state space to find a solution. This process is slow when the state space is large. By using the LLM to define subgoals, our method reduced the search area for the low-level policy. The agents only needed to learn how to reach the next nearby subgoal. This made the learning process more efficient. This result suggests that breaking down the time scale is an effective way to speed up training in large systems [21,22].

3.3. Scalability to Large Environments

We tested the performance of the method by increasing the map size from \$10 \times 10\$ to \$50 \times 50\$. The performance of the baseline models dropped as the map size increased. On the \$50 \times 50\$ map, the success rate of the QMIX model fell to below 30%. However, the L2M2

 doi:10.20944/preprints202602.0689.v1

framework maintained a success rate of over 60%. Figure 2 shows the generated paths in a grid environment with obstacles. Similar to the paths shown in the figure, our agents found valid routes by connecting subgoals and avoiding collisions [23]. This visual evidence supports the numerical data. It shows that the hierarchical structure helps agents navigate large spaces without getting stuck.

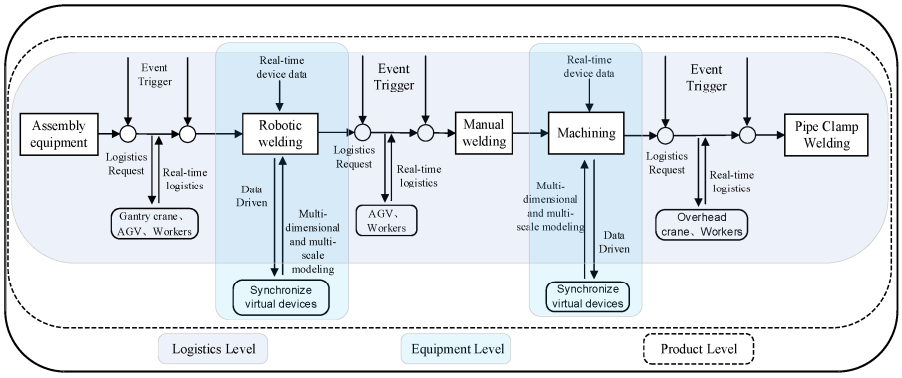

Figure 2. Visualization of planned paths in a grid environment with obstacles.

3.4. Role of the Language Model

Finally, we analyzed the effect of the language model. We compared the full model with a version that used random high-level goals. The results show that the model with random goals failed to solve the task. This shows that the logic provided by the LLM is necessary. The LLM understands the physical limits of the environment and generates logical steps. For example, it directs agents to "pick up the tool" before "repairing the machine." The low-level policy cannot learn this link easily from rare rewards. Therefore, the combination of logic and motor control is the main reason for the success of the system [24].

4. Conclusions

In this paper, we proposed a method that combines Large Language Models with multi-agent learning. The results show that this method increases the success rate by 20.5% compared to standard baselines. It also reduces the training time by 1.6 times. Unlike previous methods, our system uses language to divide long tasks into small goals. This finding shows that separating planning from control helps solve complex problems. This method is useful for applications like warehouse automation and rescue robots. However, the processing speed is slow. Future studies should use smaller models to increase the speed.

References

1. Fu, Y., Gui, H., Li, W., & Wang, Z. (2020, August). Virtual Material Modeling and Vibration Reduction Design of Electron Beam Imaging System. In 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA) (pp. 1063-1070). IEEE.
2. Alharbi, A. M., Alshehri, G., & Elhag, S. (2024, November). Reinforcement learning of emerging swarm technologies: A literature review. In Proceedings of the Future Technologies Conference (pp. 478-494). Cham: Springer Nature Switzerland.
3. Yue, L., Xu, D., Qiu, D., Shi, Y., Xu, S., & Shah, M. (2026). Sequential Cooperative Multi-Agent Online Learning and Adaptive Coordination Control in Dynamic and Uncertain Environments.
4. Abdulghani, A. M., Abdulghani, M. M., Walters, W. L., & Abed, K. H. (2024). Performance Evaluation of Multi-Agent Reinforcement Learning Algorithms. Intelligent Automation & Soft Computing, 39(2).
5. Chen, F., Liang, H., Yue, L., Xu, P., & Li, S. (2025). Low-Power Acceleration Architecture Design of Domestic Smart Chips for AI Loads.

6. Donatus, R., Ter, K., Ajayi, O. O., & Udekwe, D. (2025). Multi-agent reinforcement learning in intelligent transportation systems: A comprehensive survey. arXiv preprint arXiv:2508.20315.
7. Chen, H., Li, J., Ma, X., & Mao, Y. (2025, June). Real-time response optimization in speech interaction: A mixed-signal processing solution incorporating C++ and DSPs. In 2025 7th International Conference on Artificial Intelligence Technologies and Applications (ICAITA) (pp. 110-114). IEEE.
8. Tantakoun, M., Muise, C., & Zhu, X. (2025, July). LLMs as planning formalizers: A survey for leveraging large language models to construct automated planning models. In Findings of the Association for Computational Linguistics: ACL 2025 (pp. 25167-25188).
9. Yang, M., Wu, J., Tong, L., & Shi, J. (2025). Design of Advertisement Creative Optimization and Performance Enhancement System Based on Multimodal Deep Learning.
10. Thomas, A., Ramesh, K., & Mohan, S. Multimodal LLM Agents: Exploring LLM interactions in Software, Web and Operating Systems.
11. Peng, H., Dong, N., Liao, Y., Tang, Y., & Hu, X. (2024). Real-Time Turbidity Monitoring Using Machine Learning and Environmental Parameter Integration for Scalable Water Quality Management. Journal of Theory and Practice in Engineering and Technology, 1(4), 29-36.
12. Karataiev, O., & Shubin, I. (2023). Formal model of multi-agent architecture of a software system based on knowledge interpretation. Radioelectronic and Computer Systems, (4), 53-64.
13. Hu, W. (2025, September). Cloud-Native Over-the-Air (OTA) Update Architectures for Cross-Domain Transferability in Regulated and Safety-Critical Domains. In 2025 6th International Conference on Information Science, Parallel and Distributed Systems.
14. Mallampati, S., Shelim, R., Saad, W., & Ramakrishnan, N. (2025). Dynamic Strategy Adaptation in Multi-Agent Environments with Large Language Models. arXiv preprint arXiv:2507.02002.
15. Xu, K., Du, Y., Liu, M., Yu, Z., & Sun, X. (2025). Causality-Induced Positional Encoding for Transformer-Based Representation Learning of Non-Sequential Features. arXiv preprint arXiv:2509.16629.
16. Navneet, S. K., & Chandra, J. (2025). Rethinking autonomy: Preventing failures in AI-driven software engineering. arXiv preprint arXiv:2508.11824.
17. Tan, L., Liu, X., Liu, D., Liu, S., Wu, W., & Jiang, H. (2024, December). An Improved Dung Beetle Optimizer for Random Forest Optimization. In 2024 6th International Conference on Frontier Technologies of Information and Computer (ICFTIC) (pp. 1192-1196). IEEE.
18. Seyde, T., Lechner, M., Rountree, J., & Rus, D. (2024, October). Competitive Multi-Team Behavior in Dynamic Flight Scenarios. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 14172-14179). IEEE.
19. Gao, X., Chen, J., Huang, M., & Fang, S. (2025). Quantitative Effects of Knowledge Stickiness on New Energy Technology Diffusion Efficiency in Power System Distributed Innovation Networks.
20. Erdogan, L. E., Lee, N., Kim, S., Moon, S., Furuta, H., Anumanchipalli, G., ... & Gholami, A. (2025). Plan-and-act: Improving planning of agents for long-horizon tasks. arXiv preprint arXiv:2503.09572.
21. Mao, Y., Ma, X., & Li, J. (2025). Research on API Security Gateway and Data Access Control Model for Multi-Tenant Full-Stack Systems.
22. Korthikanti, V. A., Casper, J., Lym, S., McAfee, L., Andersch, M., Shoeybi, M., & Catanzaro, B. (2023). Reducing activation recomputation in large transformer models. Proceedings of Machine Learning and Systems, 5, 341-353.
23. Liu, S., Feng, H., & Liu, X. (2025). A Study on the Mechanism of Generative Design Tools' Impact on Visual Language Reconstruction: An Interactive Analysis of Semantic Mapping and User Cognition. Authorea Preprints.
24. Çavuş, B., & Aktaş, M. (2023). A new adaptive terminal sliding mode speed control in flux weakening region for DTC controlled induction motor drive. IEEE Transactions on Power Electronics, 39(1), 449-458.